Software Requirements Specification

for

Hiking Assistant

Version 1.1

Prepared by Yunxuan Liu, Yawen Huang and Josetta Rautapää / GROUP F

Embedded Systems Development - ELEC-E8408

Aalto University

24/3/2025

Table of contents

Table of contents 2
Revision history 4
1. Introduction 5
1.1 Purpose 5
1.2 Scope 5
1.3 Definitions, Acronyms, and Abbreviations 6
1.4 References 7
1.5 Overview 7
2. Overall Description 7
2.1 Product perspective 7
2.2 Product functions 9
2.3 User Characteristics 9
2.4 Constraints 10
2.4.1 Hardware limitations 10
2.4.2 Software dependencies 10
2.4.3 Operating Environment 10
2.4.4 Network Constraints 10
2.4.5 External dependencies 11
2.4.6 System architecture diagram 12
2.5 Assumptions and Dependencies 12
3. Specific Requirements 13
3.1 External interface requirements 13
3.1.1 Hardware Interface 13
3.1.2 Software Interface 13
3.1.3 User Interface 13
3.1.4 Communication Interface 13
3.2 Functional Requirements 13

3.3 Performance Requirements 14
3.4 Design Constraints 15
3.4.1 Operating System Compatibility 16
3.4.2 Programming Language & Frameworks 16
3.4.3 Power Constraints 16
3.4.4 Communication Protocols 16
3.4.5 Hardware Limitations 16
3.5 Software System Attributes 16
3.5.1 Reliability 16
3.5.2 Availability 17
3.5.3 Security 17
3.5.4 Maintainability 17
3.5.5 Portability 17
3.6 Other Requirements 17
4. Supporting Information 18
4.1 Appendices 18
4.2 Testing Plan 18

Revision history

Version Number	Date	Authors	Comments
1.0	21/2/2025	Group F	First draft
1.1	24/3/2025	Group F	Revised following feedback: expanded definitions, clarified scope, detailed performance constraints, added system architecture description, security protocols, and traceability structure.

1. Introduction

1.1 Purpose

The purpose of this document is to define the requirements for the Hiking Tour Assistant system. This system integrates a smartwatch and Raspberry Pi to perform three key functions: (1) real-time data collection of steps and distance, (2) processing and estimation of calories burned, and (3) visualization of hiking statistics via a web UI. This document serves as a reference for stakeholders, developers, and testers, ensuring a shared understanding of the project and aiding in development, testing, and future enhancements.

This document describes the overall system and specific requirements. It serves as a reference to stakeholders, developers, and testers, ensuring a shared understanding of the project. This foundation could be beneficial for following development, testing and future enhancements.

This document will be maintained by Group F throughout the project lifecycle and updated at each milestone. Revisions will be reflected in the revision history section.

1.2 Scope

The system consists of two main components, Smartwatch and Raspberry Pi. The smartwatch collects real-time step count and distance data using its built-in sensors and transmits it to the Raspberry Pi via Bluetooth Low Energy (BLE). The Raspberry Pi then processes this data, estimates calories burned, and provides a web-based user interface for visualization. This interaction enables users to monitor their hiking performance efficiently in real time.

The system aims to provide users with a simple way to track their hiking activity and enable them to monitor their progress over time. Helping users improve their fitness, providing a convenient and real-time overview of movement, and offering personalized insights on calories burned based on individual data.

Funding Source:
This project is funded through a pilot initiative by the Helsinki City Council in collaboration with the Finnish Hiking Tourism Board and Finnish Fitness Consortium, supporting the development of digital solutions for nature and wellness tourism in Finland.

Key Stakeholders of the Hiking Tour Assistant system include:

- **Embedded Development Team (Group F):** A small embedded systems company with experience in smartwatch platforms and embedded firmware, responsible for software and hardware integration.
- **Helsinki City Council:** Project commissioner with interest in promoting hiking and nature activities through digital tools.
- **Finnish Hiking Tourism Board & Finnish Fitness Consortium:** Expected end users and promoters of the final product for broader public and fitness tourism applications.
- **End Users:** Hikers, nature lovers, and tourists in Finland who wish to monitor their hiking performance and personal fitness.
- **Course Instructors and Evaluators (Aalto University):** Oversee development progress, evaluate deliverables, and ensure learning objectives are met.

1.3 Definitions, Acronyms, and Abbreviations

- BLE (Bluetooth Low Energy): A wireless communication protocol used for energy-efficient data synchronization between the smartwatch and Raspberry Pi.
- ESP32: A low-power microcontroller used in the LiLyGo smartwatch for data processing and Bluetooth communication.
- GPIO (General Purpose Input/Output): A set of digital pins on the Raspberry Pi used to interface with external sensors and devices.
- LiLyGo: A smartwatch model that includes an ESP32 microcontroller, used for step counting and data transmission.
- Accelerometer: A sensor that detects changes in motion and orientation, used in the smartwatch to measure step count.
- RPi (Raspberry Pi): A small single-board computer used for backend processing, data storage, and hosting the web UI.
- Linux Operating System: An open-source Unix-like OS that serves as the base platform for Raspberry Pi, offering command-line tools and package management essential for development and deployment.
- BluePy: A Python module that provides an interface to BLE using GATT protocol, supporting reading/writing Bluetooth characteristics.
- SQLite3: A lightweight embedded relational database used on Raspberry Pi to store structured hiking session data.
- Flask: A Python-based web framework used for building the web UI on the Raspberry Pi.

- **UI (User Interface)**: The graphical part of the system that allows users to interact with the application, displaying step count, distance traveled, and calories burned.
- **Authentication**: The process of verifying a user's identity before granting access to session data or system features.
- **HTML (HyperText Markup Language)**: The standard language used for creating and structuring content on the web interface.
- **Request**: A message sent by a client to initiate an action on the server.
- **RBAC (Role-Based Access Control)**: A security mechanism that restricts system functions and data visibility based on user roles.

1.4 References

- IEEE 830-1998 Software Requirements Specification Standard
- ISO/IEC/IEEE 29148-2011 System and Software Engineering - Requirements Engineering
- LiLyGo TWatch Library
- Bi-direction BLE communication between Raspberry Pi/Python (with PyQt5 GUI) and ESP32/Arduino Nano RP2040 Connect
- Omnicalculator Calories Burned Formula

1.5 Overview

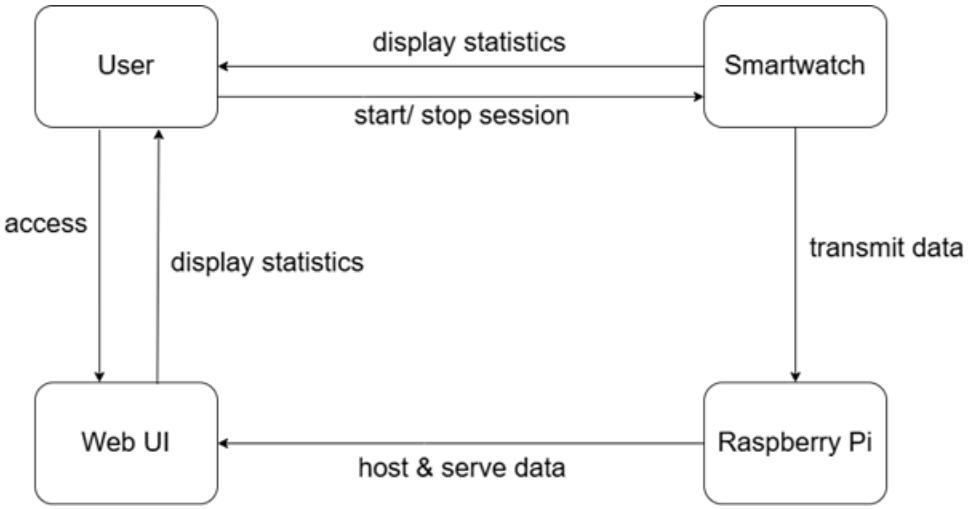
This document provides a structured overview of the system and its requirements to enhance comprehension. The following Overall Description section explains the product's perspective, functions, intended users, constraints. It also identifies assumptions and dependencies that impact development. Section 3 details the specific requirements of the system from different aspects.

2. Overall Description

2.1 Product perspective

The Hiking Tour Assistant is an embedded system developed for tracking and analyzing hiking activities. It integrates a LiLyGo smartwatch (with ESP32) and a Raspberry Pi, communicating via Bluetooth Low Energy (BLE). The smartwatch collects real-time data, and the Raspberry Pi processes, stores, and visualizes this data through a web UI.

The context diagram below shows the system's components and interactions:



The diagram defines how each system component interacts with. The key components and their roles are as follows:

User:

- Starts and stops hiking sessions using the smartwatch.
- Views hiking statistics displayed on the smartwatch screen.
- Accesses detailed statistics through the web UI.

Smartwatch:

- Receives start/stop commands from the user.
- Displays real-time statistics such as step count and distance.
- Transmits collected data to the Raspberry Pi via Bluetooth.

Raspberry Pi:

- Receives hiking data from the smartwatch.
- Processes and stores data.
- Hosts and serves the web UI for users to access their hiking statistics.

Web UI:

- Allowing users to access.
- Displays processed statistics such as step count, distance, and calories burned.

2.2 Product functions

This section outlines the key functionalities of the Hiking Tour Assistant system. The system comprises two primary hardware components: a smartwatch and a Raspberry Pi. Below is a breakdown of the functions of each component.

Smartwatch Functions:

- Start and stop Hiking sessions.
- Count real-time steps using a built-in accelerometer.
- Convert step count to walking distance.
- Display figures of step and distances on the screen.
- Communicate data to the Raspberry Pi via Bluetooth.

Raspberry Pi Functions:

- Receive data from the smartwatch over Bluetooth.
- Process data and estimate calories
- Store data
- Provide a web-based UI (user interface) to visualize hiking statistics.

2.3 User Characteristics

- Hikers and fitness enthusiasts who are seeking a wearable device for tracking hiking sessions and view real-time data.
- Tourists that are interested in tracking their daily activity levels, including step counts and movement data, for personal insights and travel records.
- Organizations and researchers that collect and analyze hiking data for research, commercial insights, or outdoor activity promotions. These users may require data export functionality and statistical analysis tools.

2.4 Constraints

2.4.1 Hardware limitations

- The system is designed to operate on Smart Wrist Watch LilyGo V3 (ESP32-based) hardware, with limited processing power, restricted memory (~520KB SRAM) and display size.
- The Raspberry Pi 3B+ serves as the primary backend processing unit, which has limited CPU and memory compared to desktop/server.
- Data transmission relies on Bluetooth 4.0 or BLE for efficient communication between components.
- No dedicated GPS or cellular module; location tracking not supported.

2.4.2 Software dependencies

- Python 3.10+ on Raspberry Pi (main backend logic)
- Flask for web server
- SQLite3 for database
- BluePy for BLE communication in the Raspberry Pi, ESP32 BLE Arduino library for BLE communication in the TTGO watch
- Arduino IDE for smartwatch firmware (C), using ESP32 board of version later than 2.0.14, with library named TTGO_TWatch_Library-master
- Linux OS (Raspberry Pi OS, Debian-based)

2.4.3 Operating Environment

- The system is intended for outdoor use, where it may be subjected to varying weather conditions and environmental factors.
- A stable Bluetooth connection between the smartwatch and the Raspberry Pi is essential for uninterrupted data synchronization.

2.4.4 Network Constraints

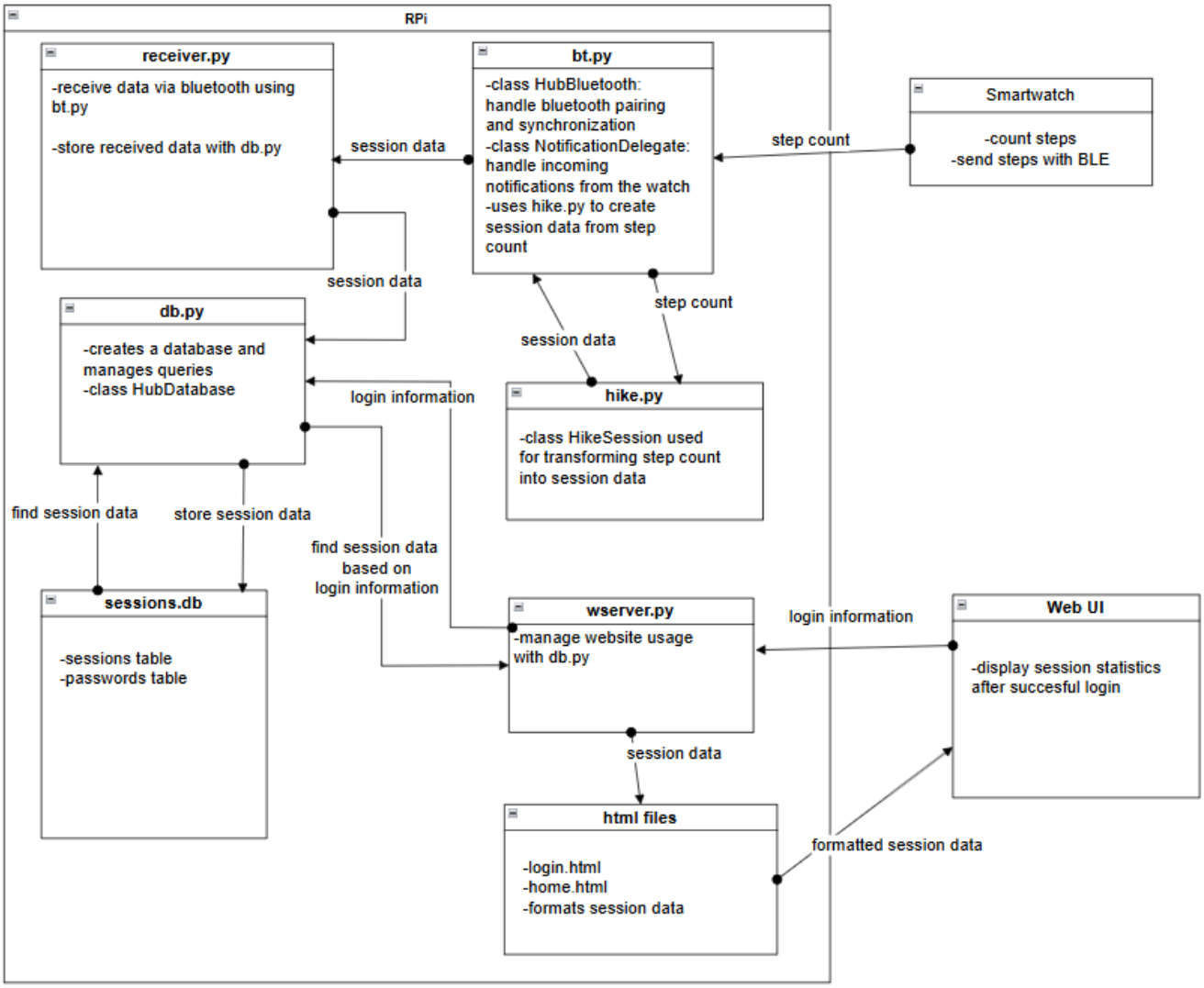
- BLE (Bluetooth Low Energy) communication is limited to short range (typically up to 10 meters) and can suffer from signal degradation due to obstacles and environmental interference.
- BLE bandwidth is limited, restricting the frequency and volume of real-time data transfers.
- Wi-Fi connectivity is required on the Raspberry Pi for users to access the web UI remotely. Weak Wi-Fi signals may result in delayed or failed data delivery.

- The system does not rely on mobile data (e.g., LTE/5G), thus real-time updates are limited to environments with Wi-Fi access.

2.4.5 External dependencies

Component	Dependency	Description
Backend Server	Flask	Python web framework
Database	SQLite3	Lightweight embedded relational database
Communication	BLE	Bluetooth Low Energy protocol to communicate
BLE Interface	BluePy	Python BLE GATT library
Firmware	Arduino IDE	Used to program ESP32 firmware
Frontend UI	HTML	User-facing web interface
OS Environment	Raspberry Pi OS	Linux-based runtime for backend

2.4.6 System architecture diagram



2.5 Assumptions and Dependencies

- The smartwatch is expected to be worn continuously during hiking sessions to ensure accurate data collection.
- The Bluetooth Low Energy (BLE) connection must remain stable between the smartwatch and Raspberry Pi for reliable data transmission.
- The Raspberry Pi must have a reliable power source, either through battery support or an external power connection, to sustain web hosting and data processing.
- Any external sensors, if used, must be compatible with Raspberry Pi GPIO or BLE interfaces to ensure seamless integration.

3. Specific Requirements

3.1 External interface requirements

3.1.1 Hardware Interface

- The smartwatch shall have a dedicated button for starting and stopping a hiking session.
- The smartwatch screen shall display real-time step count and traveled distance during the hiking session.
- The smartwatch screen shall provide success messages when system successfully uploads the information.
- The smartwatch shall include a Bluetooth module for synchronizing session data with the Raspberry Pi.
- The Raspberry Pi shall process received session data, compute burned calories, and store historical data.
- The smartwatch shall use an accelerometer sensor to detect movement and count steps accurately.

3.1.2 Software Interface

- The system shall store all hiking session data in a structured database.
- The smartwatch firmware shall support BLE communication for data exchange with the Raspberry Pi.

3.1.3 User Interface

- The web UI shall display step count, distance traveled, and calories burned in an intuitive and interactive format.

3.1.4 Communication Interface

- The smartwatch shall establish and maintain a BLE connection with the Raspberry Pi for real-time data transfer.
- The Raspberry Pi shall transmit processed session data to a web server for remote user access.

3.2 Functional Requirements

FR-01

When the smartwatch “start” button is pressed while no session is active, the system shall initiate a new hiking session.

FR-02

When the smartwatch “stop” button is pressed while a session is active, the system shall terminate the session.

FR-03

During an active hiking session, the smartwatch shall continuously record step count and display it in real time.

FR-04

The smartwatch screen shall display step count and distance traveled after the session is finished.

FR-05

After completing a hiking session, the user shall be given the option to either upload or discard the session data.

FR-06

If the user chooses not to upload the session, the session data shall be permanently deleted from the watch.

FR-07

If the user chooses to upload the session, the watch shall send the session data to the Raspberry Pi via Bluetooth, where it will be synchronized and stored.

FR-08

If Bluetooth transmission fails, the system shall allow the user to manually retry uploading the session data.

FR-09

After synchronization, the Raspberry Pi shall compute an estimate of burned calories based on user parameters.

FR-10

The web UI shall provide a summary of the last recorded sessions, including their step count, traveled distance, and calories burned.

FR-11

The web UI shall only show session data for a logged-in user.

3.3 Performance Requirements

PR-01

The smartwatch shall process only one hiking session at a time to prevent data conflicts.

PR-02

The smartwatch shall display statistics for only the current or the most recent hiking session.

PR-03

The watch shall continuously monitor for button presses, ensuring that any user input is detected without missing it, with a response time of no more than one second.

PR-04

The system shall collect, process, and display updated hiking metrics within 5 seconds of data generation.

PR-05

The web UI shall respond to user requests within 1 second under standard network conditions (Wi-Fi ≥ 10 Mbps).

PR-06

Step count accuracy shall be at least 90%, with an error margin not exceeding 5%.

PR-07

Bluetooth synchronization shall complete within 10 seconds after connection is established.

PR-08

The system shall notify the user after successful Bluetooth transmission.

PR-09

The Raspberry Pi shall respond to Bluetooth event interrupts with < 300 ms latency.

PR-10

System power consumption shall not exceed 1 W on the smartwatch and 5 W on the Raspberry Pi under typical load.

PR-11

The system shall log all data synchronization events with timestamps to support diagnostics.

3.4 Design Constraints

This section outlines the design constraints imposed by hardware limitations, software compatibility, and industry standards. These constraints ensure the system's stability, efficiency, and adaptability.

3.4.1 Operating System Compatibility

- The system shall be developed and executed on Raspberry Pi OS to ensure compatibility with hardware components and optimized performance.

3.4.2 Programming Language & Frameworks

- The backend software shall be developed using Python for ease of implementation and compatibility with Flask/Django frameworks.
- The smartwatch firmware shall be programmed using Arduino IDE to support ESP32-based hardware.

3.4.3 Power Constraints

- The smartwatch shall consume no more than 1 W during active data collection.
- Raspberry Pi power draw shall remain below 5 W during computation and transmission.
- The hardware design shall support battery or portable power sources to accommodate outdoor environments.

3.4.4 Communication Protocols

- The system shall utilize Bluetooth Low Energy (BLE) for data synchronization to minimize power consumption.

3.4.5 Hardware Limitations

- The smartwatch hardware shall include an accelerometer for step counting and a display for real-time feedback.
- The Raspberry Pi shall have sufficient processing power and storage capacity minimum of 1 GB RAM and 16 GB storage to handle incoming data and computations without performance degradation.

3.5 Software System Attributes

3.5.1 Reliability

- The system shall implement a data persistence mechanism to ensure recovery in case of power failure during data transfer.
- Data integrity checks shall be performed before storing session data to prevent corruption.

3.5.2 Availability

- The smartwatch shall remain operational whenever powered on, ensuring continuous data collection.
- The Raspberry Pi shall run a background service to manage incoming data and remain accessible for synchronization at all times.
- In case of a failed Bluetooth connection, the user shall have the option to either try data transmission again or discard the current session.

3.5.3 Security

- Access to stored session data shall require successful user authentication using a unique username and password.
- The web interface shall implement role-based access control (RBAC) to prevent unauthorized modifications.
- The login interface shall validate user credentials before granting access to protected resources. Unauthorized users shall be denied access with appropriate error messages.
- Users shall only be able to view and access their own session data.

3.5.4 Maintainability

- The system shall support over-the-air (OTA) updates for both the smartwatch firmware and Raspberry Pi software.
- Installing any software update shall not exceed 30 minutes, ensuring minimal downtime.

3.5.5 Portability

- The system shall be compatible with multiple Raspberry Pi models (3B+, 4, or 400) for flexibility.
- The smartwatch firmware shall be designed to support future ESP32-based hardware upgrades without major software modifications.

3.6 Other Requirements

- A logging mechanism shall record system events and errors for debugging purposes.
- The system should be designed with scalability in mind, allowing future integration with additional sensors or cloud-based storage if required.

4. Supporting Information

4.1 Appendices

Development Tools:

- PyCharm or VSCode for Python development.
- Flask for the backend web UI.
- Arduino IDE for smartwatch firmware development.
- SQLite for local database management.

Hardware Recommendations:

- Raspberry Pi 3B+ for computational tasks.
- External accelerometer for step counting (if additional sensors are needed).
- Portable LCD screen for real-time data display.

4.2 Testing Plan

- Validate sensor integration to ensure accurate step counting and distance calculations.
- Test SQLite database performance to confirm session data persistence.
- Ensure the web interface provides responsive and accurate data visualization.
- Perform stress testing to verify system performance under different user loads.